VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2018 Dan Novák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dan Novák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Argutec, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Mgr. Jiří Dvorský, Ph.D.**

Konzultant bakalářské práce: Michal Holiš

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.	prof. Ing. Pavel Brandštetter, CSc.
vedoucí katedry	*děkan fakulty*

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 13. dubna 2018 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 13. dubna 2018 .....

Argutec, s.r.o.
Na Nivách 1339/4
70030 Ostrava-Zábřeh

IČ: 29453721
http://www.argutec.eu

Rád bych poděkoval společnosti Argutec, s.r.o, jmenovitě Ing. Michalu Holišovi, CTO, který mě k odborné praxi přijal, za jeho cenné rady a vedení v celém průběhu práce. Dále také děkuji všem zaměstnancům firmy za přijetí do kolektivu, za vstřícný přístup a ochotnou pomoc, kdykoliv bylo potřeba.

Abstrakt

Tato práce si klade za cíl popsat a shrnout průběh vykonávání individuální odborné praxe ve firmě Argutec, s.r.o. na pozici programátor C#/.NET. Věnuje se popisu práce a řešení dílčích úkolů, které byly studentovi zadány. Je psána chronologicky dle postupu práce na projektu. V textu je rovněž možnost se seznámit s jednotlivými technologiemi, které byly použity. Závěrem je srhnut přínos působení ve firmě Argutec, s.r.o. pro studenta.

Klíčová slova: C#, .NET, Argutec, s.r.o., Individuální odborná praxe, Windows Forms, Programování, Grafické uživatelské rozhraní, Raspberry Pi

Abstract

The aim of this thesis is to describe and summarize the work performed when undergoing an individual professional practice in Argutec, s.r.o., as C#/.NET programmer. It is devoted to the description of the work and the solution of its constituents, as they were assigned to the student. It is written chronologically, according to the order in which the work on the project was performed. The reader can also get acquainted with specific technologies which were used. Finally, there is a summary of how beneficial the practice in the company Argutec, s.r.o. was for the student.

Key Words: C#, .NET, Argutec, s.r.o., Individual professional practice, Windows Forms, Programming, Graphical User Interface, Raspberry Pi

Obsah

Seznam použitých zkratek a symbolů **8**

Seznam obrázků **9**

Seznam výpisů zdrojového kódu **10**

1 Úvod **11**

2 O společnosti **12**
2.1 Oblasti působení 12
2.2 Služby a produkty 12

3 Použité nástroje a technologie **14**
3.1 C# 14
3.2 ASP.NET Core 14
3.3 Team Foundation Server 14
3.4 JavaScript 14
3.5 GTK# 15
3.6 Windows Forms 15
3.7 Raspberry Pi 3 15
3.8 Bash 15
3.9 Python 15

4 Zaškolení a zadání projektu **16**
4.1 Seznámení se s konvencemi a vnitropodnikovými procesy 16
4.2 Představení projektu 16
4.3 Všeobecné zadání 16
4.4 Odhad časové náročnosti úkolů 17

5 Řešení **18**
5.1 Implementace v ASP.NET Core 18
5.2 GTK# 19
5.3 Windows Forms 20

6 Závěr **30**
6.1 Uplatněné znalosti a dovednosti 30
6.2 Chybějící znalosti a dovednosti 30
6.3 Shrnutí 31

Literatura **32**

Seznam použitých zkratek a symbolů

GTK	–	GIMP Toolkit
GUI	–	Graphical User Interface
B2B	–	Business To Business
TFS	–	Team Foundation Server
LCD	–	Liquid Crystal Display
PIN	–	Personal Identification Number
XML	–	Extensible Markup Language
GPIO	–	General Purpose Input/Output
API	–	Application Programming Interface
WiFi	–	Wireless Fidelity
ESSID	–	Extended Service Set Identifier
IP	–	Internet Protocol
DHCP	–	Dynamic Host Configuration Protocol
USB	–	Universal Serial Bus

Seznam obrázků

1 Grafické rozhraní 17
2 Třídní diagram modulového systému 21
3 Grafické rozhraní fotogalerie 28

Seznam výpisů zdrojového kódu

1 Přijímání a zpracování zpráv obsahující znaky 22
2 Metoda pro zpracování vyvolané události 23
3 Posílání zpráv v závislosti na obsahu bufferu 24
4 Testování připojení zásláním pingu na Google server 27
5 Vytváření složek a pořizování snímků 27

1 Úvod

Tato bakalářská práce se zabývá průběhem absolvování indiviudální odborné praxe ve firmě **Argutec, s.r.o.** Příležitost pracovat v uvedené společnosti se naskytla díky vypsání pozice v informačním systému KatIS. Zaujala mě možnost pracovat na pozici programátor C#/.NET, a tak jsem tuto firmu oslovil a po absolvování pohovoru a zpracování vstupního testu jsem byl přijat na praxi.

Na počátku mým hlavním úkolem bylo se naučit používat a znát firemní konvence, později jsem ovšem již vstoupil do pracovního provozu tím, že jsem byl pověřen prací na projektu, ve kterém figuruje automatická schránka na zásilky, resp. jednočipový počítač, který box ovládá. V tomto projektu jsem využil několik technologií před konečným rozhodnutím pro Windows Forms, ve kterém jsem vyvíjel GUI a backend funkcionalitu aplikace. Dále bylo úkolem uvést aplikaci do chodu na jednočipovém počítači, Raspberry Pi, s postupným dodáváním nových možností použití.

V první části jsem představil firmu, ve které jsem praxi vykonával. V následující části jsou stručně popsány technologie, které byly v projektu použity. V dalších kapitolách se již věnuji samotným úkolům, které mi byly zadány k vypracování, samozřejmě i s jejich řešením.

Každému zpracovanému projektu vyhrazuji prostor úměrně ke skutečnému času, který jsem nad ním strávil. Na závěr shrnuji, které absolvované předměty byly důležité k úspěšnému vykonávání praxe a hodnotím, jakým přínosem pro mě celá stáž byla.

2 O společnosti

Argutec, s.r.o. je mladá společnost, založena v roce 2012, která se zabývá vývojem průmyslových systémů strojového vidění, kamerových a termovizních systémů a měření odchylky produktu od jeho ideálního tvaru. Firma má dále zkušenost s oblastmi vývoje jako např. operátorské aplikace pro obsluhu výrobních linek, aplikace pro vizualizaci dat, aplikace pro statistické a analytické vyhodnocení archivovaných dat, matematické modely a optimalizace procesů řízení výroby, propojení s řídicími systémy linek, obohacení stávajících systémů o snímače, čítače a jiné s možností propojení s dalším software, zajištění řídicích systémů, aplikace termovize v průmyslu, instalace kamerových systémů a záznamových zařízení.

Největší doménou společnosti, jak již její název, který je odvozen od řeckého vševidoucího obra Arguse Panoptese, naznačuje, je strojové vidění. Jedná se tedy především o zpracování obrazu z kamer a následné vyhodnocení výsledků. Je využíváno například ve výrobě pro zjišťování vad na produktech, upozornění obsluhy stroje na chybu, kontrola počtu výrobků, úprava nastavení výrobní linky. Dále může posloužit u bezpečnostních kamer k identifikaci osob či vozidel. Člověk se svým subjektivním pohledem je tedy nahrazen objektivním posouzením aplikace. [1]

2.1 Oblasti působení

Argutec, s.r.o. působí především v oblasti systémů strojového vidění, termovizí a provádí instalaci monitorovacích IP kamer. Velkou roli v produktech rovněž hraje automatizace ve výrobě.

V neposlední řadě vyvíjí softwarová řešení na míru (např. B2B aplikace, objednávkové a výrobní systémy, aplikace pro operátory výrobních linek a jiné), což je sféra působení, do které jsem se na odborné praxi zapojil i já.

Společnost Argutec, s.r.o působí také ve vědě a výzkumu na projektech. Kupříkladu pro VŠB-TUO, pro kterou vyvíjí elektroniku, firmware a analytický software duralové kapsle a čidla smyku pro logování fyzikálních veličin při průchodu sypkým materiálem v zásobníku.[1]

2.2 Služby a produkty

Surface Scan System Systém slouží k monitorování kvality výrobků v reálném čase a k analýze dat. Vyhodnocuje povrchové vady, dokáže přesně identifikovat jejich pozici a přináší tak možnost přizpůsobit výrobní operace či vyřadit výrobek z produkce.

Shape Scan System Řešení, které za pomocí systému laserů a kamer dokáže rozhodnout o rovinnosti vyráběného produktu. Díky němu lze tedy rozpoznat, zda je určitý výrobek zvlněn, poškozen nebo zda se na jeho povrchu nenachází nežádoucí předmět, např. nečistota. Systém se používá hlavně při výrobě kolejnic, lahví a dalších produktů, u nichž hraje rovinnost jejich povrchu významnou roli.

Cracks Detection System Tento systém se skládá ze sady kamer a systému osvětlení. Jeho hlavní funkcí je detekce trhlin na produktech neprůsvitného materiálu. Původně byl vyvinut pro použití v automobilovém průmyslu pro kontrolu lisovaných dílů karosérie, kde hrozí riziko vzniku trhliny (např. v oblasti svarů). Systém je dodáván včetně aplikace, kde může operátor sledovat detekování a být informován o případných nálezech. V aplikaci lze také prohlížet historie a dále vyhodnocovat archivované informace o trhlinách společně s obrazovými daty.

Furnace Detection System Furnace Detection System slouží k monitorování prostoru v peci při teplotách do 1400°C. Kamera je nainstalována na plášť pece a umožňuje operátorovi sledovat dění v peci. Systém lze doplnit například o automatizovaný monitoring počtu kusů, který prošel pecí či provádět jinou analýzu obrazu. Pokud jsou k systému přidány termovizní kamery, lze sledovat i průběh teplotního pole materiálu.

Steel and Slag Detection System Termovizní monitorovací systém, který slouží k rozpoznávání strusky a kovu v tekutém likvidu. Při napojení na řídící systémy je možno například vyvolat alarm či zastavit/naklonit pec jako reakce na nastalou situaci. Hlavním přínosem produktu je tedy zvýšení kvality oceli, minimalizace přenosu strusky a prevence před únikem oceli.

Legování oceli Komponenta vyvinutá na platfomě .NET, využívající evoluční algoritmy, která slouží k legování oceli. Po dodání informací o oceli (např. její hmotnost, aktuální analýza, požadovaná jakost) a parametrů legur (složení, cena) je komponenta schopná vypočítat výsledné množství legur dosažení požadované jakosti oceli a výslednou cenu za legury.

Průmyslové počítače Firma také distribuuje vysoce výkonné průmyslové počítače, ať už jako součást některého z výše zmíněných produktů, ve kterých hrají významnou roli, anebo je prodává samostatně, dle přání a požadavků zákazníků.

3 Použité nástroje a technologie

Na praxi jsem měl příležitost pracovat s několika technologiemi. S většinou jsem se již ve škole v nějaké podobě setkal, ale některé pro mě byly nové. Nejvíce jsem používal vývojářské prostředí Microsoft Visual Studio 2017, které jsem potřeboval k vývoji v jazyce C#. Součástí VS 2017 byl také verzovací systém Team Foundation Server, jehož základní funkce jsem se musel naučit zvládat. K vývoji jsem nejprve používal ASP.NET Core, ve kterém jsem dostal první významnější úkol. Bylo také nezbytné využít JavaScriptu pro psaní skriptů pro správnou funkcionalitu webové stránky na straně klienta. Později jsem byl pověřen nastudováním a použitím multiplatformních knihoven GTK#, ve kterých jsem měl vyvinout grafické rozhraní. Protože ovšem hlavní vývoj mířil na jednočipový počítač Raspberry Pi a klasická Windows Forms aplikace se jevila jako lepší variantou, tak bylo rozhodnuto o setrvání u zmíněné platformy. Pro psaní jednoduchý krátkých pomocných programů jsem použil Bash a Python.

3.1 C#

C# je typově bezpečný, objektově-orientovaný jazyk, který vychází z jazyka C++ a Javy. Pochází z dílny společnosti Microsoft a je neustále ve vývoji. Lze v něm psát webové, konzolové, databázové nebo formulářové aplikace, webové či Windows služby.[2]

3.2 ASP.NET Core

Jde o vysoce výkonný, multiplatformní framework, který slouží zejména k tvorbě webových aplikací a služeb, tvorbě mobilních back-endů. Verze .NET Core 2.0 vyšla 14. srpna 2017 a šlo ji tedy v době počátku mé praxe na podzim 2017 považovat za novinku. Tato platforma je díky snaze Microsoftu zajištěna kvalitní dokumentací, což velice usnadnilo hledání informací.[3]

3.3 Team Foundation Server

Team Foundation Server umožňuje týmovou spolupráci při vývoji softwaru a je přenositelný do různých vývojových prostředí. V Argutec, s.r.o. byla tedy možnost jej použít jako součást Visual Studia 2017. Poskytuje zejména privátní úložiště kódu a verzovací systém, do kterého členové týmu ukládají své verze kódu určitého projektu. Díky tomu je možné patřičně reagovat v případě konfliktu, což může díky archivaci verzí znamenat i návrat ke starší verzi v případě nutnosti.[4]

3.4 JavaScript

JavaScript je nenáročný, interpretovaný, objektově orientovaný jazyk, který je využíván zejména ve webovém prostředí, kde zajišťuje interaktivitu s webovou stránkou na straně klienta. Používán je například ve formulářích k usnadnění vyplňování a zajišťuje tak tedy větší uživatelský komfort.[5]

3.5 GTK#

Jde o multiplatformní soubor nástrojů a knihoven, který slouží k vytváření grafických rozhraní. GTK# je nástavba nad základním projektem gtk+, která umožňuje použití nástrojů s jazykem C#. Jeho výhodou je, že jej lze použít jak v prostředí Windows, tak díky platformě Mono je možné kompilovat programy využívající GTK# i na jiných operačních systémech.[6]

3.6 Windows Forms

Windows Forms, zkráceně také WinForms, je knihovna, která umožňuje tvorbu grafického uživatelského rozhraní a je součástí .NET Frameworku Microsoftu. Od jeho použití se upouští kvůli nahrazení modernějším nástrojem Windows Presentation Forms. Aplikaci je rovněž možné spouštět na platformě Mono.[7]

3.7 Raspberry Pi 3

Jedná se o jednočipový počítač o velikosti kreditní karty, který byl původně navržen pouze pro využití ve vzdělávání. Obsahuje základní hardware počítače a obvykle je používán s operačním systémem Raspbian, který je založený na Debianu. Uživateli poskytuje mnoho možností, jako například psaní textu, hraní her či sledování videí, ale nedisponuje tak výkonnou výpočetní silou, aby mohl konkurovat stolním počítačům. Naopak je ale snadné doplnit moduly jako například LCD obrazovku, fotoaparát či senzory, které je možné použít k zobrazování či sběru dat, které můžeme použít pro nejrůznější aplikace.[8]

3.8 Bash

Bash je standardním interpretem příkazů v Linuxu, který poskytuje rozhraní mezi uživatelem a systémem. Obvykle je také používán pro psaní skriptů, ve kterých je možné využít základních algoritmických konstrukcí. Namísto použití kompilovaného jazyka lze v některých případech považovat Bash za dostatečnou náhradu pro vykonání určitých úkolů.[9]

3.9 Python

Je to dynamicky interpretovaný, objektově-orientovaný jazyk, který používá čistou a jednoduchou syntaxi. V projektu byl použit jako úhlednější alternativa k Bashi. Zajistil v něm drobnou práci se soubory a v malé míře i volání systémových procesů.[10]

4 Zaškolení a zadání projektu

4.1 Seznámení se s konvencemi a vnitropodnikovými procesy

Důležitou součástí mého vstupu do pracovního procesu bylo seznámení se s chodem firmy. Na úvod jsem tedy měl za úkol se seznámit s dokumentací, která popisuje konvence psaní kódu, dále také s diagramem popisujícím jakým způsobem se postupuje při zpracovávání úkolů apod. Získal jsem také přístup do Timetrackeru, což je online aplikace, která umožňuje přehledné zaznamenávání výkonu práce každého zaměstance. Díky ní jsem v průběhu celé praxe měl přehled o tom, kolik hodin pracuji a čím jsem se u daného úkolu zabýval.

Pro osvojení návyků jsem byl tedy pověřen přepracováním a zlepšením aplikace, která mi byla předložena k řešení u vstupního testu.

4.2 Představení projektu

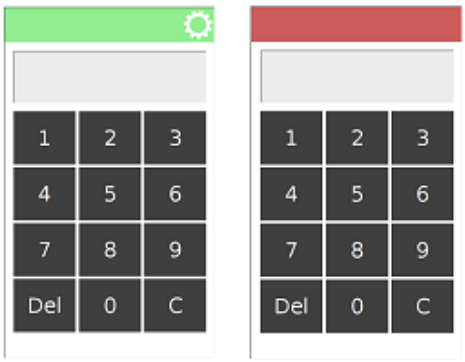
Po celou dobu trvání praxe mým hlavním zadáním bylo vyvinout aplikaci pro Raspberry Pi pro obsluhu automatického výdejního boxu. Schránka se skládá ze samotné konstrukce, jednočipového počítače Raspberry Pi s LCD obrazovkou, reproduktorem, fotoaparátem a také elektromagnetickým zámkem, který je napojen na relé a externí zdroj.

4.3 Všeobecné zadání

Aplikace má především poskytnout grafické rozhraní, jehož prostřednictvím bude moci uživatel schránku ovládat a na jednotlivé vyvolané akce bude upozorněn grafickými a zvukovými vodítky. Aplikace by měla taktéž obsahovat logiku pro otevírání či zavírání schránky a zajišťovat komunikaci se zámkem, který je k Raspberry Pi připojen přes GPIO (General Purpose Input/Output) piny. Z bezpečnostních důvodů bude aplikace opatřena fotoaparátem, který po zadání PINu pořídí snímky a uloží je do galerie.

Pro konfiguraci celého produktu je také nutné zajistit uživateli možnost nastavení sítě – bezdrátové i ethernetové – a galerii pro prohlížení pořízených fotografií. Uživatel bude moci nastavovat připojení pouze v aplikaci, minimálně v této verzi, a proto je důležité myslet na všechny případy a zajistit bezproblémovou konektivitu.

Poslední zpracovanou částí bylo umožnění naskenování čárového kódu jako alternativa k zadávání PIN kódu.



Obrázek 1: Grafické rozhraní

4.4 Odhad časové náročnosti úkolů

Úkol	**Časová náročnost [h]**
Zaškolení	16
Implementace v ASP.NET Core	40
GTK#	80
Windows Forms	264

5 Řešení

5.1 Implementace v ASP.NET Core

V prvním kroku jsem obdržel zadání zpracovat stránku pro přihlášení v .NET Core 2. Jelikož jsem s touto architekturou tehdy ještě nepracoval, úkol zahrnoval studium možností a seznámení se s MVC (Model View Controller), ve kterém jsem projekt založil.

5.1.1 Přihlašovací stránka

Prostudoval jsem alternativy a nakonec jsem zvolil autentizaci za pomocí cookies. Bylo také potřeba vyřešit automatické přesměrovávání nepřihlášeného uživatele po zobrazení kterékoliv stránky projektu a okamžité přesměrování na hledanou stránku po zadání správného loginu a hesla. Uživatelské jméno i heslo jsem měl zvoleno pouze fixně, jelikož jsem s projektem teprve začínal a bylo nutné otestovat funkčnost přihlašování.

Po vytvoření modelu pro login jsem přidal do původní konfigurační třídy *Startup* autentizaci za pomocí cookies a do třídy pro přihlašování jsem napsal asynchronní metodu pro login s akcí HTTP POST. Zde jsem také zajistil ověření správných přihlašovacích údajů a vyřešil přesměrovávání na adekvátní stránku po stisku tlačíka pro přihlášení.

5.1.2 Nastavení sítě

Dalším krokem bylo vytvořit sekci pro nastavování ethernetové sítě. Měl jsem reprodukovat formulář nastavování sítě z operačního systému Windows, kde se zadává IP adresa, maska podsítě, výchozí brána, upřednostňovaný a alternativní DNS server a možnosti volby, zda chceme získat IP nebo DNS adresu automaticky anebo, po zadání zmíněných hodnot, je získáme staticky. Dále jsem na stránku měl přidat zaškrtávací políčko, které dává možnost volby, jestli chceme takto nakonfigurovat i bezdrátové připojení.

Po tvorbě tříd modelů, které jednoduchou strukturou reprezentovaly všechny konfigurační možnosti, a kontroleru pro nastavení sítě, jsem si nechal automaticky vygenerovat pohled za pomocí scaffoldingu. Pro to, aby se v daném pohledu kurzor při zadávání adres pohyboval mezi jednotlivými oktety, jako při nastavení IP adresy v prostředí Windows, jsem využil nalezenou aplikaci, napsanou v jazyce JavaScript, používající knihovnu jQuery, kterou jsem změnil dle potřeby. Aby uživatel nemohl ukládat chybné IP adresy, bylo potřebné vytvořit třídu, která za pomocí regulárního výrazu sloužila k validaci vložených hodnot.

Následovalo řešení samotného ukládání, které jsem implementoval prostřednictvím XML serializace. Všechny modelové třídy jsem učinil serializovatelnými a po použití jednoduchých metod jsem tedy ukládal nastavení do souboru. Za pomocí JavaScriptu jsem nastavil sekci tak, že automaticky načítala již uložené hodnoty a voláním metody deserializace se hodnoty ze souboru vkládaly do jednotlivých polí a oktetů.

Posledním krokem bylo zabezpečit validaci na straně serveru, nastavit patřičná hlášení uživateli v případě chybných nebo chybějících hodnot.

5.2 GTK#

V projektu jsem dostal za úkol pokračovat s knihovnami a nástroji GTK#, aby se aplikace mohla použít v prostředí Raspbianu. Od implementace v ASP.NET Core se tedy pro vyvíjenou aplikaci upustilo, a tak jsem ji začal vyvíjet nanovo.

5.2.1 Studium

Protože ve firmě nikdo neměl zkušenosti s použitím GTK#, byl jsem odkázán na dokumentaci Mono Project. V první fázi jsem tedy trávil čas objevováním možností tvoření GUI a seznamováním se s tím, jak je možné GTK# zkompilovat a zobrazit v prostředí Windows 10.

Následně jsem si již měl představit a adaptovat grafické rozhraní pro Raspberry Pi v určitém rozlišení. Vyvíjel jsem tedy dva formuláře: jeden sloužil k zobrazení nastavení sítě, stejně jako v předchozí podkapitole, a druhý k zobrazení klávesnice, jejímž účelem bylo poskytnout možnost zadání IP adresy do formuláře nastavení sítě.

Tvorba fomulářů byla časově náročná, protože jsem v GTK# nenašel způsob, jak do nich vkládat prvky bez potřeby vkládat jeden prvek do druhého. Například pro získání možnosti vyvolání akce při stistku tlačítka jsem musel vložit každý obrázek do boxu, který to umožňuje. Každý box jsem musel v kódu zařadit do správné buňky tabulky a tabulku připojit k hlavnímu oknu aplikace. Pro usnadnění orientace jsem byl nucen dodržovat dobré praktiky psaní kódu a dopomoci si vysvětlujícími komentáři a regiony, ale i tak se mi jevil tento způsob tvorby GUI bez grafické ukázky v editoru jako velmi těžkopádný.

Do formuláře s numerickou klávesnicí jsem doplnil logiku pro posuv kurzoru při psaní či mazání čísel v polích, pro zadávání jednotlivých oktetů IP adresy. Využil jsem také modelů z prostředí .NET Core pro umožnění serializace a deserializace nastavení připojení z/do XML souboru. Rovněž jsem uplatnil třídy pro validaci a serializaci, které jsem napsal dříve.

5.2.2 Reorganizace projektu a zprovozňování GTK# na Raspberry Pi

Pro udržení pořádku v projektu a vyhnutí se redundanci kódu jsem se po konzultaci s kolegou rozhodl vytvořit knihovnu, která bude obsahovat všechny modelové a pomocné třídy, které využívám v .NET Core i GTK# aplikaci.

Následně jsem obdržel Raspberry Pi a byl s ním seznámen. Rozhraní, které jsem vytvořil za pomocí GTK#, jsem se snažil zprovoznit na Raspberry Pi, ale vyskytly se neočekávané chyby. Po konzultaci s vedoucím, Ing. Holišem, se od použití GTK# ustoupilo.

5.3 Windows Forms

Prvním úkolem bylo zjistit, jak funguje WinForms na Raspberry Pi a jaké jsou v něm odlišnosti v zobrazení oproti platformě Windows. Vytvořil jsem tedy jednoduchou aplikaci, kde jsem si otestoval vykreslování základních prvků (např. tlačítek, textů) a bylo učiněno rozhodnutí, že budu projekt tedy vyvíjet s použitím Windows Forms. Tato část praxe byla nejdelší, protože jsem postupně dostával nové úkoly, které jsem do projektu měl zakomponovat.

5.3.1 Tvorba grafického rozhraní

Na počátku bylo potřeba navrhnout nové GUI pro zadávání PINu. Raspbian dodal tlačítkům nežádoucí styl, který pocházel z konfigurace operačního systému, a proto jsem vytvořil nové *UserControl*, kde jsem překryl metodu *OnPaint* a nastavil patřičné vlastnosti pro správné zobrazení.

Pro plynulé přepínání mezi formuláři jsem založil hlavní formulář, který byl pouze MDI kontejner a slouží tedy jako rodič každého celostránkového formuláře. Po uvedení do provozu Raspberry Pi se ovšem vyskytly nedostatky. Největším problémem bylo chvění obrazu při vykreslování nových formulářů a dlouhá doba odezvy. Po poradě jsem nastavil vlastnosti *DoubleBuffered* kladnou hodnotu a překryl jsem taktéž vlastnost *CreateParams*. Počítač ale zřejmě není ideálně vyladěn na takové rychlé vykreslování a podobné úlohy mu trvají déle, a tak jsem po zkoumání možností vyzkoušel některé formuláře načíst ještě před zobrazením a zobrazit je jen v případě, když jsou uživatelem vyžadovány. Tento pokus se zdařil a používal jsem tuto techniku dále, kdykoliv bylo potřeba.

Posledním zásadním problém byly nesprávné rozměry aplikace na Raspberry Pi. Když jsem tvořil GUI v prostředí Windows 10, tak jsem nastavil veškeré hodnoty přesně tak, aby se správně zobrazily na Raspberry Pi, ale nestalo se tomu tak. Určité rozměry a pozice bylo třeba přenastavit podle potřeby z důvodu odlišného vykreslení.

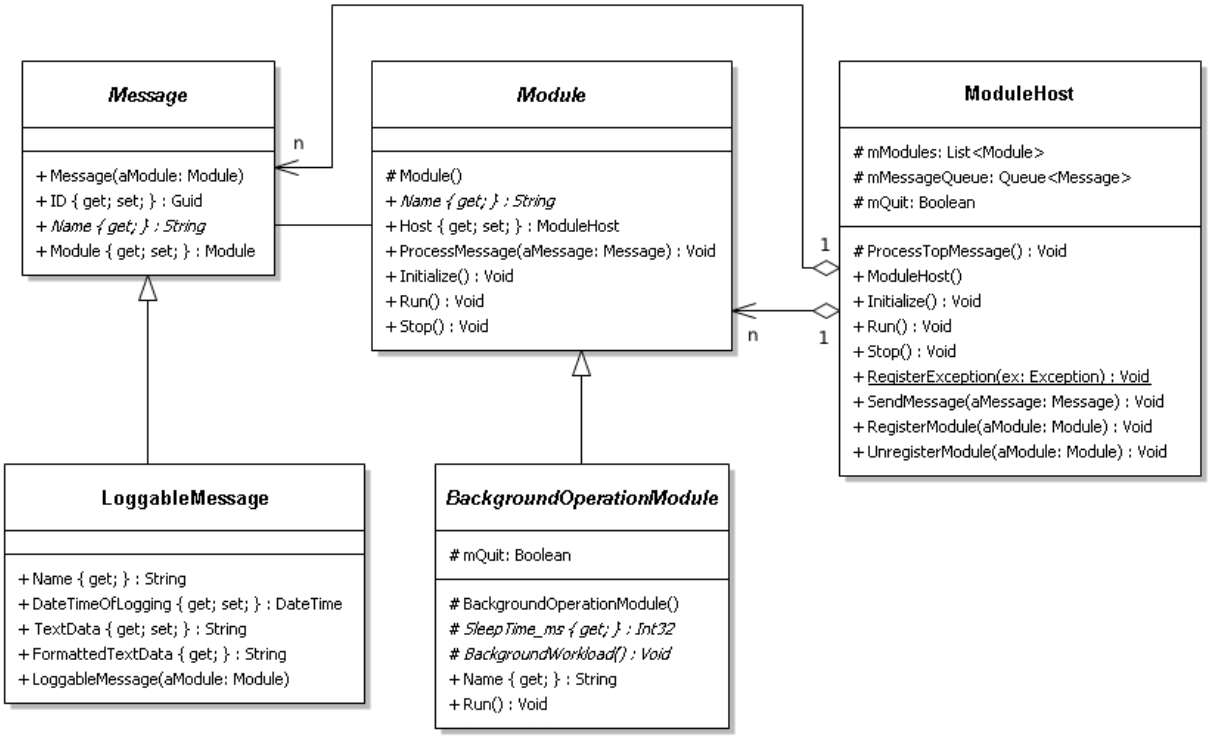
5.3.2 Modulový systém

Zásadní částí projektu je modulový systém, který zajišťuje veškerou komunikaci v aplikaci prostřednictvím zpráv. Tento systém vytvořil můj vedoucí, a proto se dále budu detailněji zaobírat až částmi, které jsem doplňoval já.

Základ tvoří knihovna, ve kterém jsou definovány abstraktní třídy *Module*, *ModuleHost* a *Message*. Třída *Module* je vzorem pro každý modul, který z ní dědí. Obsahuje například vlastní identifikaci, metodu pro zpracování zprávy a vlastnost typu *ModuleHost*, která registruje model k určitému hostiteli. *ModuleHost* tedy zaopatřuje veškerou komunikaci mezi zaregistrovanými moduly. Obsahuje seznam modulů, frontu zpráv, které jsou moduly posílány a také metody pro spravování modulů. Je zde také mimo jiné implementována metoda pro zpracování a rozeslání zpráv, která je periodicky spouštěna a využívá zámku kvůli bezpečnosti při paralelním vykonáním kódu za pomocí bazénu vláken. Abstraktní třída *Message* pouze definuje, jak vypadá zpráva,

která bude v systému posílána. Identifikuje ji za pomocí identifikačního čísla, jména a obsahuje vlastnost typu *Module*, kam ukládáme modul, který zprávu vyvolal.

Na následujícím třídním diagramu jsou zobrazeny třídy *Module*, *ModuleHost* a *Message*, které tvoří základ systému. Tříd, které dědí z *Message* anebo *Module* je příliš mnoho pro tento diagram, a proto zde představu uvádím jeho zjednodušenou podobu s pouze dvěmi dědícími třídami, *LoggableMessage* a *BackgroundOperationModule*



Obrázek 2: Třídní diagram modulového systému

5.3.2.1 Posílání PINu po znacích Pro zajištění základní funkcionality aplikace, tzn. ověřování správnosti PINu, mým úkolem bylo implementovat nové zprávy a moduly po vzoru jiných, které mi můj vedoucí, Ing. Michal Holiš, připravil v další knihovně v projektu.

Vytvořil jsem tedy zprávu *PinKeyPressedMessage*, obsahující datum a čas stisku tlačítka, výchozí modul a samotný znak, který je posílán. Třída dědí z třídy *OperationResultMessage*, která umožňuje ukládání času operace a ukládá také typ výsledku, který je definován jako enumerátor.

Dále bylo nezbytné vytvořit modul, který zprávu bude posílat. Ten obsahuje pouze identifikaci modulu a metodu pro zpracování znaku, která daný znak pošle ve zprávě, o které píši výše.

Posledním krokem bylo vytvořit třídu, *PinResolverModule*, ve které zprávy přijímám a pracuji s nimi.

```
public override void ProcessMessage(Message aMessage)
{
   base.ProcessMessage(aMessage);

   if (aMessage is PinKeyPressedMessage)
   {
      PinKeyPressedMessage lPinKeyPressedMessage = aMessage as
          PinKeyPressedMessage;
      if (lPinKeyPressedMessage != null)
      {
         SaveIntoBuffer(lPinKeyPressedMessage.PressedKey);
         TimeOfLastInput = DateTime.Now;
      }
   }
   // ...
}
```

Výpis 1: Přijímání a zpracování zpráv obsahující znaky

5.3.2.2 Interní buffer a posílání výsledku

PinResolverModule obsahuje logiku pro práci se znaky. Pokud příchozí znak reprezentuje akci smazání posledního znaku anebo vymazání všech znaků, je učiněna příslušná akce. V každém jiném případě jsou znaky ukládány do pole typu *char*. Jestli je buffer naplněn (PIN obsahuje čtyři znaky), je PIN zkontrolován oproti správnému PINu a je vyvolána událost s výsledkem (typu enumerátor) a zadaným PINem. Následně je obsah bufferu smazán.

Dále je také implementována metoda, která periodicky počítá čas od přijetí posledního znaku a pokud je doba delší než definovaná hodnota, je obsah bufferu promazán, aby neúplné zadávání PINu nezasahovalo do následujícího používání aplikace.

5.3.2.3 Implementace v grafickém rozhraní

Pro umožnění práce s pouze jednou instancí typu *ModuleHost* jsem vytvořil v projektu WinForms aplikace statickou třídu *ModuleContext*, ve které jsem inicializoval a zaregistroval všechny moduly, které využívám. Dále jsem vytvořil metodu pro spuštění hostitele a poskytl metody, jejichž prostřednictvím mohu volat metody modulů, například pro poslání zprávy obsahující znak stisknutého tlačítka.

V hlavním formuláři bylo nutné spustit hostitele ve vlákně odděleném od hlavního GUI aplikace, ať je možné se vyhnout blokaci programu, s pomocí třídy *BackgroundWorker*.

Za použití metody *BeginInvoke* jsem ve formuláři s klávesnicí čekal na vyvolanou událost a napsal metodu, která má řešit příchozí výsledek.

```
private void PinResolverModule_PinEntered(object sender, P17035.ModuleSystem.
    CommonModules.EventArgs.PinEnteredEventArgs e)
      {
         BeginInvoke(new MethodInvoker(() =>
         {
               ResolvePinResponse(e.Result == P17035.ModuleSystem.CommonModules.
                  Messages.OperationResultMessage.ResultEnum.Success);
         }));
   //...
         textBoxPin.Clear();
      }
```

Výpis 2: Metoda pro zpracování vyvolané události

Posledním krokem pro dokončení komunikace bylo vytvořit nový *UserControl*, který slouží jako notifikační panel. Dle přijaté zprávy zobrazí uživateli, zda zadaný PIN je správný tak, že zabarví panel zeleně, resp. červeně.

5.3.3 Přidání zvuků

Byl mi přidělen malý reproduktor, který je možné zapojit do Raspberry Pi, aby uživatelé dostávali i zvukovou informaci, že například stiskli tlačítko anebo jestli byl jejich zadaný PIN správný či nesprávný.

Po vyzkoušení, že v prostředí Raspbianu nefunguje správně třída *SoundPlayer*, která by mohla být pro takový případ zvolena, jsem vytvořil novou třídu, která funkcionalitu zajišťuje.

U této příležitosti jsem se poprvé setkal se spouštením procesů v terminálu na Raspberry Pi a tuto techniku jsem vydatně využíval v projektu i nadále.

Pro přehrání audia jsem tedy volal aplikaci *aplay* prostřednictvím terminálu a po získání vhodných souborů formátu *wav*, jsem mohl použití zvuků v programu demonstrovat.

5.3.4 Hlavní funkce schránky - zámek

Můj vedoucí mě pověřil zajištěním jedné z hlavních funkcí schránky, což je schopnost zamykat a odemykat zámek. Předal mi elektromagnetický zámek výrobce LOCKPOL a elektromagnetické relé, které funguje jako spínač. Jelikož Raspberry Pi není schopné samo o sobě dodávat tak veliký proud pro sepnutí zámku, bylo nutné zapojit na výstup relé, pomocí něhož jsme zámek spínali napětím z externího zdroje. Díky pomoci zkušenějšího kolegy se vše podařilo zprovoznit a funkčnost obvodu jsme ověřili vykonáním několika příkazů v terminálu, které zajistily přístup k použitému GPIO pinu, nastavily směr a poslaly hodnotu 1 nebo 0. Při nastavení pinu na kladnou hodnotu se rozsvítila dioda na relé a zámek byl otevřen.

Následujícím krokem tedy bylo umožnit aplikaci ovládat GPIO piny a reagovat tak na uživatelem zadaný PIN. V projektu, který mi byl svěřen, již ovšem existovala knihovna, volně dostupná na internetu, která zmíněnou práci do jisté míry abstrahuje a obstarává veškerou práci s piny, včetně hospodaření s pamětí.

Součástí projektu byl také modul *GpioModule*, který přijímá zprávy typu *GpioWriteRequest*. Za pomocí výše zmíněné knihovny pak zapisuje hodnoty, nesené ve zprávě, do příslušných pinů.

Mým úkolem tedy bylo použít modul v projektu. Ve třídě *PinResolverModule*, ve které je řešeno ověřování správnosti PINů a výsledky jsou posílány do grafického rozhraní, jsem vytvořil instanci zprávy typu *GpioWriteRequest*, která obsahuje požadované změny a posílá zprávy přes hostitele typu *ModuleHost*. Nutnou změnou bylo, že nyní již aplikace musela být spouštěna pouze s příkazem *sudo*, aby práce s GPIO piny byla možná. Neoprávněný uživatel nemá právo měnit jejich stavy.

```
private void SendResultMessage(char[] aBuffer)
{
   string lBufferString = new string(aBuffer);

   Messages.OperationResultMessage.ResultEnum lResult = OperationResultMessage.
       ResultEnum.Error;
   GpioWriteRequest.GPIOPinWriteChange lGpioWriteChange = new GpioWriteRequest.
       GPIOPinWriteChange()
   {
      Pin = GPIO.GPIOPins.GPIO_21,
      Value = false
   };

   if (IsPinCorrect(lBufferString))
   {
      lResult = OperationResultMessage.ResultEnum.Success;
      lGpioWriteChange.Value = true;
      mTimerLocker.Start();
   }
   OnPinEntered(lResult, lBufferString);
   ClearBuffer();

   Host.SendMessage(new GpioWriteRequest(this, new GpioWriteRequest.
       GPIOPinWriteChange[]
   {
      lGpioWriteChange
```

```
  }));
}
```

Výpis 3: Posílání zpráv v závislosti na obsahu bufferu

5.3.5 Zapojení do sítě

Schránka má komunikovat s webovým API, přijímat správné PINy nebo například odesílat zprávy o různých akcích. Bylo tedy důležité implementovat možnost, aby se uživatel mohl přihlásit na internet, ať už prostřednictvím WiFi anebo ethernetu. Ačkoliv je vcelku snadné síť nakonfigurovat v prostředí Raspbianu, bylo nutné myslet na to, že uživatelé finálního produktu budou odkázáni pouze na aplikaci, aby jim tedy poskytla všechny nezbytné možnosti nastavování. Je pouze zvažováno do budoucí verze produktu, že bude existovat i možnost nastavení pro pokročilé uživatele, kde by jim přístup do operačního systému byl umožněn.

5.3.5.1 WiFi Jako první jsem dostal za úkol řešit bezdrátové připojení. Zahrnovalo to hledání dostupných sítí v okolí a možnost připojení se ke zvolené síti za pomocí hesla. Nejdříve jsem neúspěšně hledal knihovnu, která by zajistila zmíněné funkce a která by zároveň fungovala i na Raspberry Pi. Musel jsem se tedy opět spolehnout na řešení problému prostřednictvím terminálu, ve kterém jsem z aplikace spouštěl procesy.

Prvním příkazem, který jsem použil, byl *iwlist wlan0 scan*, z jehož výsledku jsem si vybral zobrazení ESSID všech dostupných sítí, neboli identifikátorů WiFi sítě, a také pro zjištění, zda je Raspberry Pi připojeno k některé z vypsaných sítí. Dále jsem z něj dokázal zjistit, která síť je zabezpečená a která ne. Výsledky jsem ukládal do objektu *WifiConnection*, které jsem uložil do seznamu dostupných sítí.

Vytvořil jsem nový formulář s prvkem *ListView*, ve kterém jsem všechny nalezené sítě zobrazoval s informací, jestli je počítač k některé z nich připojen. Bylo také nutné zajistit aktuálnost dat, a proto jsem do třídy formuláře doplnil časovač s úkolem spouštět výše zmíněný proces v intervalu a získávat tak nové informace o dostupných sítích.

Pro samotné připojení jsem vyhledal konfigurační soubor, kde si Raspberry Pi ukládá sítě, ke kterým se dříve snažilo připojit. Každý záznam se zabezpečenou sítí obsahuje její identifikátor a heslo. Pro úspěšné připojení k určité síti je tedy zapotřebí zapsat nová data do tohoto souboru a pak vykonat příkaz pro rekonfiguraci sítě.

Problém jsem vyřešil tak, že jsem si nejdříve načetl obsah souboru do seznamu objektů typu *WifiConnection*, který jsem po žádosti uživatele doplňoval anebo měnil v závislosti na provedené akci v hlavním formuláři. Pokud se chce uživatel připojit k dané síti, tak musí zadat heslo, které je do seznamu následně uloženo. Veškeré změny v seznamu se dále promítnou v konfiguračním souboru.

Dále bylo nutné vykonat příkaz pro rekonfiguraci sítě, aby operační systém zaregistroval změny v souboru a příslušně obnovil aktuální připojení. Pokud proces vrátil kladný výsledek, tak

byla spuštěna metoda pro periodické zjišťování, zda je počítač již připojen k dané síti. Časovač v určitém intervalu vykonával příkaz, jehož výstupem bylo jméno sítě, ke které se Raspberry Pi připojilo, pokud se operace zdařila. Uživatel pak byl uvědoměn o úspěšném připojení za pomocí události. V opačném případě časovač vypršel a ke spojení nedošlo.

Velmi podobným způsobem jsem postupoval i při doplňování možnosti připojení se k nezabezpečené síti, ale neměl jsem doposud příležitost funkčnost otestovat.

Práce na tomto úkolu byla časově náročná, hlavně kvůli nutnému studiu problematiky konfigurace bezdrátového spojení v prostředí Raspbianu, ale také z důvodu, že jsem zmíněné procesy již nemohl spouštět a testovat v operačním systému Windows 10, kde jsem aplikaci vyvíjel.

5.3.5.2 T9 klávesnice

Uživatel bude také mít možnost připojení klávesnice ke schránce přes port USB. Je očekáváno, že jen zřídka bude potřeba zadávat heslo pro přihlášení se na WiFi síť, ale i tak pro větší komfort uživatelů se po konzultaci se zadavateli projektu dospělo k rozhodnutí, že by bylo vhodné vytvořit T9 klávesnici. Původně jsem byl pověřen nalezením vhodné virtuální klávesnice, která by svými rozměry splnila funkci i na dotykové obrazovce, ale nepodařilo se mi takovou najít.

Implementoval jsem tedy nový formulář, kde jsem za pomocí časovačů napodobil funkcionalitu T9 klávesnic. Při dlouhém doteku je vypsáno číslo a při krátkém stisku uživatel prochází polem znaků z daného tlačítka.

Tato klávesnice neobsahuje všechny symboly, které by v heslu mohly být použity, a tak se jedná spíše o doplněk, který ovšem ve většině případů postačí.

5.3.5.3 Připojení na Ethernet

Box by měl zajišťovat možnost i kabelového připojení k internetu a bylo tedy nutné vytovřit nový formulář a třídu pro konfiguraci adres.

Z předchozích úkolů jsem použil třídy pro validaci, serializaci a také samozřejmě třídy reprezentující jednotlivé prvky konfigurace sítě jako například IP či DNS adresy. Vytvořil jsem tedy opět obdobný formulář ve Windows Forms, umožnil oboustrannou komunikaci s XML souborem, který uchovává konfiguraci a napodobil jsem tak dle zadání formulář z prostředí Windows.

Vytvořil jsem také další formulář s číselnou klávesnicí pro možnost zadání IP adresy do některého z nabízených polí v nastavení, pokud si zvolíme statickou konfiguraci a nepoužijeme tedy protokol DHCP.

Na pozadí bylo nutné zajistit, obdobně jako u bezdrátového připojení, práci s konfiguračním souborem a volání procesů, například pro zjištění stavu spojení. Objekt *NetworkSettings*, jenž obsahuje veškeré nastavení sítě z formuláře, používám po stisku tlačítka pro uložení ve třídě *EthernetConnector*, abych mohl nové nastavení zapsat do konfiguračního souboru. Řádky souboru se načítají do pole a podle potřeby jsou doplňovány, měněny anebo mazány. Záleží na tom, jestli uživatel preferuje statické nastavení anebo automatické.

Následně je spouštěn Python skript, který restartuje všechny potřebné procesy pro restartování připojení, aby systém zaregistroval provedené změny v konfiguračním souboru. Po zjištění

jména ethernetového rozhraní příkazem *ip addr flush dev [rozhraní]* vyčistíme současné nastavení IP adresy na příslušném rozhraní pro umožnění opětovného připojení.

Posledním krokem je spuštění procesu připojování a periodické ověřování stavu připojení. Za pomoci jednoduchého Bash skriptu jsem z používaného rozhraní v intervalu posílal ping na server Googlu. Pro tuto volbu jsem se rozhodl z důvodu spolehlivosti daného serveru.

```
#!/bin/bash
if ping -I $1 -q -c 1 -W 1 8.8.8.8 >/dev/null; then
  echo "up";
else
  echo "down";
fi
```

Výpis 4: Testování připojení zásláním pingu na Google server

V průběhu celého procesu jsou vyvolávany události, které uživateli zobrazují informativní formuláře s aktuálním stavem připojení.

5.3.6 Pořizování snímků

Aby byla schránka lépe zabezpečená, je v plánu do ní vestavět malý fotoaparát, který bude pořizovat snímky při zadání správného nebo nesprávného PINu a také při skenování čárových kódů. Fotografie budou pořízeny tři a to v intervalu dvou sekund. Snímky se následně budou ukládat do galerií, které si uživatel může prohlížet v aplikaci.

Byl mi tedy přidělen malý fotoaparát, který se dá jednoduše zapojit na základní desku Raspberry Pi a mým úkolem bylo implementovat dané zadání. Nejdříve jsem zjistil, jakým způsobem se fotoaparát ovládá a napsal jsem krátký Bash skript, který ze všeho nejdříve vytvoří složku s aktuálním datem, časem a informací o správnosti nebo nesprávnosti PINu. Rozhodl jsem se pro tento postup, protože bylo nutné vymyslet způsob, jak přehledně zorganizovat snímky do složek, aby je bylo možné jednoduše zobrazovat v aplikaci. Ve skriptu jsem také spustil příkaz k focení společně s nastavením časových údajů, názvu snímku a velikosti fotografií. Proces *raspistill* se postaral o všechno ostatní na pozadí a tři nové fotografie s popiskem času v horní části se zobrazily v dané složce.

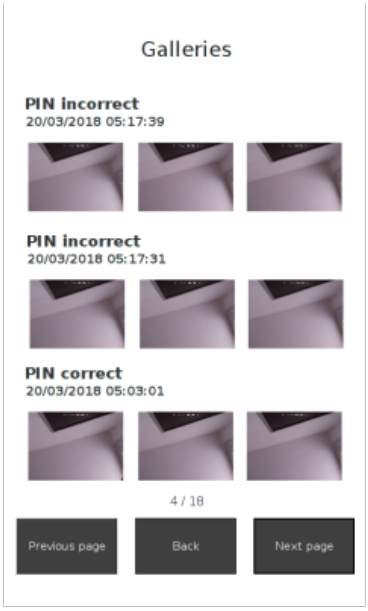
```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M%S")
mkdir -p images/$DATE-$1
raspistill -t 4000 -tl 2000 -vf -a 12 -o $(pwd)/images/$DATE-$1/img%02d.jpg -w
    1024 -h 768 -n
```

Výpis 5: Vytváření složek a pořizování snímků

5.3.6.1 Galerie Dalším krokem tedy bylo vytvořit nový formulář, kde by měl uživatel možnost galerie procházet. Původně jsem řešení vyvíjel způsobem, že jsem do formuláře vložil prvek *ListView*, do kterého jsem chtěl nahrát náhledy, čas založení galerie a událost, která pořízení fotografií vyvolala. Byla by pak možnost se dostat na další formulář, který by zobrazil všechny snímky v dané galerii. Vedoucí ovšem vznesl námitku, že kdyby těch fotografií byl velký počet, tak by Raspberry Pi mimořádně dlouhou muselo načítat celý seznam. Nakonec se tedy dospělo k rozhodnutí, že vytvořím formulář se třemi objekty typu *GalleryControl*, který dědí z *UserControl*, přičemž každý objekt bude obsahovat snímky dané galerie a všechny další vyžadované informace. Každý obrázek pak lze zobrazit v detailní podobě v novém formuláři.

Dané řešení je vhodné z důvodu, že je načítáno minimální množství dat a aplikace tedy není zpomalována špatnou prací s pamětí. Ve spodní části formuláře *GalleryDisplay* jsou tlačítka pro přepínání mezi stránkami, aby bylo možné zobrazit galerie, které byly pořízeny dříve. Dodal jsem také číselný ukazatel, na jaké stránce z kolika se uživatel právě nachází.

Ve třídě hlavního formuláře jsem tedy zajistil načítání jmen všech složek, které splňují předpis určitého regulárního výrazu ve složce dedikované pro galerie. Pro každý *GalleryControl* na stránce pak volám jeho metodu *LoadImagesIntoGalleryControl*, která podle názvu dané složky vyhledá podle regulárního výrazu všechny fotografie, načte je do objektů typu *PictureBox* a nastaví patřičné informativní popisky. Dále se také třída *GalleryControl* stará o uvolňování paměti, ke kterému dochází při každé změně obrázků – tzn. v případě, že uživatel postoupí na další stránku, je galerie třeba obměnit.



Obrázek 3: Grafické rozhraní fotogalerie

5.3.6.2 Čtečka čárových kódů

Následujícím úkolem bylo umožnit uživatelům skenovat čárové kódy a umožnit tak přístup do schránky. Funkcionalita tedy má být velmi podobná zadávání číselného PINu. Čtecí zařízení jsem ovšem neměl k dispozici, a tak jsem byl pověřen simulací zadávání kódu za pomocí klávesnice.

Použil jsem opět stejnou formu komunikace, tzn. posílání zpráv, s tím rozdílem, že sekvence znaků nebyla posílána postupně, ale v jednom celku. Čtečka posílá znak nového řádku po načtení všech znaků kódu, a tak byl původní plán poslat kód po stisknutí klávesy Enter. Na Raspberry Pi byl ale problém, že tuto akci intepretoval tak, že má zopakovat poslední úkon, který byl provede. Jinak stisknutí této klávesy vůbec neregistroval. Můj vedoucí tedy navrhl, abych tuto problematiku řešil způsobem, že nastavím krátký časovač, který po např. 200 ms sekvenci zprávu pošle. Problém byl takto prozatímně vyřešen, ale zatím nebyl testován v praxi se skutečnou čtečkou.

5.3.7 Pokračování projektu

Projekt v době psaní této práce pokračuje dále a jsou stále doplňovány nové funkcionality. Aby bylo možné se schránkou komunikovat přes internet, je nutné implementovat posílání zpráv na webové API, které budou obsahovat její stav či vyvolané akce o tom, co se s boxem v dané chvíli děje. Předpokládám tedy, že to bude hlavním krokem k vytvoření prototypu produktu, který již bude splňovat veškeré základní požadavky. Budoucnost projektu se bude odvíjet především dle zadavatele práce a vedoucího projektu ze společnosti Argutec, s.r.o.

6 Závěr

6.1 Uplatněné znalosti a dovednosti

Na odborné praxi jsem využil mnoho znalostí a dovedností, které jsem získal v průběhu studia na vysoké škole.

Zásadním předmětem, po jehož absolvování jsem byl motivován k hledání praxe s daným zaměřením, byl Programovací jazyky II, ve kterém jsem přišel do styku s prostředím .NET a jazykem C#. Naučil jsem se v něm také základy Windows Forms, které jsem měl příležitost uplatnit. K uvedenému předmětu se také nedílně pojí předmět Architektura technologie .NET, který je vyučován v 5. semestru. V něm jsem se také naučil nepostradatelné věci, které jsem v průběhu praxe denně využíval.

Mezi předměty, které mě připravily na základní programovací návyky a algoritmické přemýšlení, bych mohl zařadit také Algoritmy I a II či Programování I a II, kde jsem byl seznámen především s programováním v jazyce C++ a s objektově orientovaném programováním, které jsem samozřejmě v jazyce C# neustále uplatňoval.

Protože jsem na praxi pracoval v prostředí Raspbianu, který je založen na Linuxu, důležité pro mě byly také vědomosti nabyté v předmětech Architektury počítačů a paralelních systémů a Operační systémy, kde jsem se naučil používat příkazovou řádku v prostředí Linuxu.

Počítačové sítě pro mě rovněž byly podstatným předmětem, protože jsem v práci na projektu řešil konfiguraci bezdrátového či ethernetového připojení a manipulaci s IP adresami.

6.2 Chybějící znalosti a dovednosti

Díky všeobecnému základu, který mi škola poskytla, jsem se nesetkal s mnoha nedostatky, co se znalostí a dovedností týče. Věci, kterými jsem se zabýval během praxe byly většinou nad rámec absolvovaných předmětů, ale bylo snadné doplnit potřebné znalosti skrze samostudium, právě díky průpravě získané v průběhu vysokoškolského studia. Příkladem by mohla být práce s Raspberry Pi, která je vcelku specifická, ale díky výše zmíněným předmětům není neznalost použití tohoto počítače velkou překážkou.

Existují ovšem dvě zásadní praktické oblasti, o kterých bych si přál získat povědomí již na půdě školy. První oblastí je základní znalost verzovacích systémů jako například TFS nebo Git, které umožňují týmovou kolaboraci na projektech. Považuji nyní schopnost používat zmíněné pomůcky za kriticky důležitou pro veškerou programátorskou činnost, kterou nebudu provozovat sám, a tudíž za zásadní předpoklad pro ucházení se o zaměstnání.

Dalším bodem je schopnost efektivně odhalovat chyby v kódu a využití integrovaných nástrojů ve vývojářském prostředí programu Visual Studio. Daná činnost patří k jedné z nejtypyčtějších výplní času při vývoji a lze ji tedy také považovat za zásadní dovednost.

K naučení těchto praktik je samozřejmě možné dojít přirozeně v průběhu samotné praxe, ale pro lepší připravenost na potenciální pracovní poměr by bylo dle mého lepší o těchto věcech získat přehled v rámci předmětů, jejichž náplň je zaměřením podobná.

6.3 Shrnutí

Absolvovaná praxe mi pomohla uvést nabyté znalosti a dovednosti z vysoké školy do profesního života. Získal jsem širší přehled o různých technologiích a cenné zkušenosti s prací ve firmě. Díky často individuální práci na úkolech jsem se naučil přemýšlet samostatně a dívat se na použitelnost vyvíjené aplikace z pohledu zákazníka. Byla ode mne tedy očekávána kreativní činnost a poskytnuta volnost v implementaci. Díky radám a trpělivosti mého vedoucího, Ing. Michala Holiše, jsem mohl zlepšit vlastní práci a přijít do styku s efektivními postupy tvorby softwaru, se kterými jsem se během studia nesetkal.

Společnost Argutec, s.r.o. mi umožnila pracovat na zajímavém projektu a poskytla mi tak možnost získat lepší představu o možnostech mého uplatnění po studiu. Hodnotím tedy tuto příležitost jako velice pozitivní a jako vhodnou volbu bakalářské praxe.

Literatura

[1] Argutec s.r.o.: Argutec s.r.o. [online]. [cit. 2018-04-04]. Dostupné z: http://www.argutec.eu/home

[2] Microsoft Docs: A Tour of C# - C# Guide [online]. [cit. 2018-04-04]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[3] Microsoft Docs: Introduction to ASP.NET Core [online]. [cit. 2018-04-04]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/

[4] Team Foundation Server: Agile, Git, CI with TFS [online]. [cit. 2018-04-04]. Dostupné z: https://www.visualstudio.com/cs/tfs/

[5] MDN: About JavaScript - JavaScript [online]. [cit. 2018-04-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[6] Mono: GtkSharp [online]. [cit. 2018-04-04]. Dostupné z: http://www.mono-project.com/docs/gui/gtksharp/

[7] Microsoft Docs: Windows Forms [online]. [cit. 2018-04-04]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/framework/winforms/

[8] Raspberry Pi FAQs: Frequently Asked Questions [online]. [cit. 2018-04-04]. Dostupné z: https://www.raspberrypi.org/help/faqs/

[9] AbcLinuxu: BASH [online]. [cit. 2018-04-04]. Dostupné z: http://www.abclinuxu.cz/clanky/navody/bash-i

[10] Python. Python 3.6.5 documentation: General Python FAQ [online]. [cit. 2018-04-04]. Dostupné z: https://docs.python.org/3/faq/general.html